# KONSEP DATA dan INFORMASI

*Oleh : Aqwam Rosadi K, SKom., MM.*

Pengantar :

Dilandasi oleh rasa keprihatinan yang mendalam karena masih banyak mahasiswa peserta sidang sarjana yang masih belum dapat menjelaskan apa itu data dan informasi di komputer untuk mata ujian yang berkenaan dengan komputer, seperti Sistem (Manajemen) Basis Data, Struktur Data, Sistem Informasi Manajemen, Berkas dan Akses, dan sejenisnya, maka saya memberi catatan berikut ini.

## ARTI KATA "DATA"

Kata "data" berasal dari bahasa Yunani "*datum*" yang berarti fakta, dan di dalam kamus bahasa Inggris ditulis dengan "*data*". "Data" yang digunakan dalam bahasa Indonesia berasal dari bahasa Inggris tersebut, tetapi harus diingat, "*data*" dalam bahasa Inggris sudah bersifat majemuk, karena tidak ada kata "*datas*" dalam bahasa Inggris. Sehingga tidaklah tepat bila kita menuliskan kata data yang dimajemukkan, seperti data-data, kumpulan data, dan sejenisnya.

Bila kita akan mengungkapkan sekumpulan data, tentulah harus dipilah-pilah tipe-tipe atau jenis-jenis datanya terlebih dulu. Misalkan kumpulan data mahasiswa dan data dosen, kumpulan data karyawan dan data konsumen, dan sebagainya.

## DATA PADA MANUSIA

Data diterima manusia karena adanya stimulus-stimulus yang dapat "ditangkap" oleh indera manusia dan dibawa oleh simpul-simpul saraf yang pada akhirnya diolah oleh otak. Tidak semua stimulus yang ditangkap indera manusia akan direkam secara permanen di dalam memori otak, sehingga muncul istilah *short term memory* (STM) dan *long term memory* (LTM).

STM bersifat sementara, misalkan kita diminta untuk menjawab pertanyaan berapa orang yang berbaju merah yang kita jumpai pada hari ini. Data STM biasanya sangat tidak kita hiraukan atau tidak pedulikan, biarkan hal itu berlalu begitu saja. Sedangkan untuk LTM, data itu sangat kita perhatikan sehingga perlu untuk kita ingat. LTM akan semakin baik menempati memori permanen bila makin sering digunakan atau diingat-ingat. Misalkan saja, bila kita diberi nomor telepon oleh seseorang yang kita pedulikan, maka semakin sering kita menghubungi nomor telepon itu, maka akan semakin mudah mengambil kembali dari ingatan.

Semakin sering ingatan tersebut digunakan, maka simpul-simpul saraf yang jumlahnya miliaran di otak dapat meraih data di memori permanen melalui berbagai jalur yang dibentuknya. Menurut penelitian, banyak sel saraf yang mati setiap harinya, sehingga bila hafalan jarang dilakukan maka bisa jadi ada simpul saraf yang mati atau rusak pada jalur raih data tersebut, dan berakibat kita menjadi lupa. Itulah perlunya media catatan tambahan di luar memori utama manusia (otak), seperti buku, kaset, dan sebagainya,

DO NOT COPY

sehingga, bila kita lupa, kita dapat mempelajarinya lagi dan menyimpan kembali di otak melalui jalur yang lain.

Ada dua jenis data yang dapat diterima manusia, yaitu jenis data yang tersurat, dan jenis data yang tersirat. Jenis data yang tersurat adalah berbagai stimulus yang secara nyata dapat ditangkap indera manusia, sedangkan jenis data yang tersirat hanya dapat "dibaca" oleh naluriah manusia, yang berkaitan erat dengan pengalaman batiniah seseorang.

## DATA DI KOMPUTER

Data di komputer berjenis data tersurat, artinya, komputer akan mengerti data yang diberikan kepadanya bila data tersebut dapat dikodekan atau dilambangkan sesuai dengan kaidah-kaidah yang sudah dimengerti oleh komputer.

Karena adanya kaidah-kaidah tentang penulisan data di komputer, maka kata "data" di komputer harus dispesifikkan karena masih bersifat luas. Data di komputer memiliki ukuran yang lebih spesifik. Satuan data terkecil di komputer adalah bit, yaitu sinyal-sinyal elektronik yang dilambangkan dengan 0 dan 1. Nilai nol diberikan bila rangkaian listrik yang menerima data tidak dialiri sinyal listrik, sedangkan nilai satu diberikan bila rangkaian listrik penerima data dialiri sinyal listrik. Rangkaian digital (dari komputer digital) tersebut menerima sinyal listrik sebesar 5 volt.

Karena sinyal listrik bersifat abstrak, maka untuk mempelajarinya, sinyal-sinyal itu diberi lambang 0 dan 1, dari sana kemudian dikenal istilah *binary digit* (bit) atau bilangan berbasis dua. Selanjutnya, untuk lebih memperingkasnya, bit-bit tersebut digabung 3 bit-3 bit menjadi bilangan oktal (berbasis 8), atau menjadi 4 bit-4 bit menjadi bilangan *hexadecimal* (berbasis 16).

Komputer yang dibuat oleh berbagai pabrik membuat kesepakatan-kesepakatan untuk melambangkan sebuah huruf atau angka, meski tidak semua pabrik setuju bersatu untuk melambangkannya, sehingga ada yang sepakat dengan kode ASCII (*American Standard Code for Information Interchange*), ada yang setuju dengan kode EBCDIC (*Extended Binary Code Decimal Interchange Code*), dan sebagainya. Misalkan, huruf 'A' di ASCII adalah kumpulan bit '10100001', tetapi di EBCDIC '11000001'.

Kumpulan bit di atas menghasilkan sebuah kode untuk huruf atau angka atau karakter lain yang disebut dengan *byte*. Sehingga *byte* adalah kumpulan dari *bit*. Satuan kapasitas penyimpanan data di komputer juga sering menggunakan *byte*. Misalkan disket dapat menampung data sebesar 1,4 *Megabyte*, dan sebagainya. 1 *Megabyte* setara dengan $2^{10}$ = 1024 bit.

Kumpulan *byte*, misalkan karakter 'A', 'L', dan 'I' yang digabung dan memiliki satu pengertian yaitu sebuah nama 'ALI', gabungan itu akan membentuk sebuah *field*. Untuk menampung data tersebut, diberikan sebuah variabel atau *field name*, sedangkan isi datanya disebut dengan nilai data (*data value*).

DEMO COPY

Misalkan *field name* = 'NAMA', dan *data value* = 'ALI'. Di dalam struktur data, setiap variabel atau *field name* harus ditentukan jenisnya. Ada jenis variabel numerik untuk menyimpan data yang berupa angka (bilangan), misalkan A = 10, ada juga variabel *string* untuk menyimpan data yang berupa huruf, atau gabungan huruf dan tanda baca lainnya, misalkan B$ = "BADU", ada juga variabel logika untuk menyimpan data yang bersifat *boolean*, misalkan X = .T., dan sebagainya. Tipe data numerik dibagi lagi menjadi *integer* dan *real*, yang digunakan sesuai dengan nilai datanya, apakah bilangan bulat atau desimal, dan sebagainya. Dan berbagai jenis dan tipe data lainnya yang biasanya sudah disiapkan (*reserved word*) oleh setiap bahasa pemrograman. Bila kita menggunakan bahasa pemrograman PASCAL, maka tentulah kita harus mendeklarasikan tipe dan jenis data di setiap variabel sebelum kita memasukkan nilai datanya.

*Field-field* yang dikumpulkan untuk memberi gambaran sebuah rangkaian keterhubungan di antara mereka terhadap sesuatu objek disebut dengan *record*. Misalkan untuk menggambarkan sebuah data seorang mahasiswa, maka kumpulan *field* NPM, NAMA, KELAS, ALAMAT, TGL_LAHIR dari seorang mahasiswa, disebut dengan *record*.

Kumpulan *record*, misalkan untuk menyimpan data mahasiswa per jurusan yang berisi 1000 orang, disebut dengan *file*. Skemanya dapat digambarkan sebagai berikut :


Gambar 1. Hubungan *field, record,* dan *file*

## BASIS DATA *(DATA BASE)*

Banyak definisi mengenai basis data, namun pada intinya, basis data adalah kumpulan *file* yang saling berelasi dan diorganisasi pada suatu media penyimpanan. Seperti dikatakan semula, bit adalah data, *field* adalah data, sedangkan kata "data" di dalam komputer harus disebutkan secara spesifik, maka basis data adalah kumpulan *file*. *File* terdiri atas berbagai klasifikasi, ada *master file, transaction file, report file, work file, program file, dump file, library file, history file,* dan sebagainya.

Keseluruhan *file* tersebut dalam basis data diorganisasikan oleh yang disebut DBMS (*data base management system*). Itulah mengapa dalam *data base*, memori yang dibutuhkan lebih besar dari *file* tradisional.

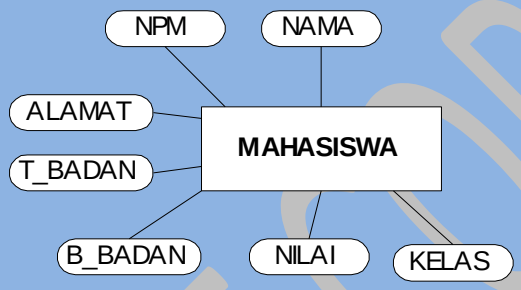
Karena basis data adalah kumpulan *file* yang saling berelasi, maka salah satu materi dalam basis data adalah ERD (*entity relational diagram*), yaitu suatu diagram yang

DO NOT COPY

menggambarkan relasi antarentitas. Entitas adalah kata lain dari *file*. Jadi, bila di suatu media penyimpanan ada data MAHASISWA, ada data DOSEN, ada data MATA_KULIAH, ada data UANG_KULIAH, dan sebagainya, hal itu belum dikatakan basis data bila data tersebut saling terpisah atau belum berelasi.

Sebelum kita lanjutkan bagaimana ERD menggambarkan relasi antarentitas, terlebih dulu, mari kita kuatkan konsep dari entitas (*file*) dan *field* (atribut). Dalam teori perancangan sistem, entitas dilambangkan dengan persegi panjang, dan atribut dilambangkan dengan elips.

Perhatikan gambar berikut ini :




Gambar 2. Ada atribut yang keliru di sebuah entitas

Digambarkan, entitas MAHASISWA memiliki atribut-atribut :

NPM (sebagai kunci utama/ *primary key*)
NAMA
ALAMAT
T_BADAN (tinggi badan)
B_BADAN (berat badan)
NILAI
KELAS

Tampak sekilas, semua atribut sudah tepat merupakan atribut dari Mahasiswa, tetapi bila lebih diamati, ada beberapa kejanggalan. Atribut dapat dikatakan sebagai suatu karakteristik dari entitas, atribut merupakan bagian dari entitas. Banyak yang mengatakan bahwa tinggi badan dan berat badan bukan merupakan atribut dari mahasiswa karena tidak pernah diproses. Apakah betul demikian ?.

Karena atribut merupakan bagian (yang disandang) dari entitasnya, maka untuk menjawab mana yang bukan atribut dari mahasiswa di atas, tanyakan saja kepada (semua) mahasiswa. Bila mahasiswa bisa menjawab secara langsung, maka ia merupakan atributnya, dan bila tidak dapat menjawab atau malah berbalik tanya, maka ia bukan atributnya.

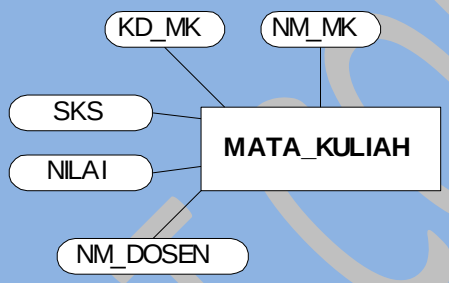
Berapa NPM anda ?
Siapa NAMA anda ?
Di mana ALAMAT anda ?

DO NOT COPY

Berapa T_BADAN (tinggi badan) anda ?
Berapa B_BADAN (berat badan) anda ?
Berapa NILAI anda ?
KELAS berapa anda ?

Pastilah untuk pertanyaan berapa nilai anda, si mahasiswa tidak dapat menjawabnya, bahkan akan bertanya, nilai (mata kuliah) apa ?. Maka, atribut (*field*) NILAI bukan (tidak boleh) merupakan atribut dari MAHASIWA. Adapun tinggi badan dan berat badan, betul data tersebut tidak diperlukan sehingga dari pada memenuhi isi memori lebih baik dibuang saja, tetapi kedua *field* tersebut adalah atribut dari mahasiswa yang sah (tetapi bersifat *optional*).

Perhatikan gambar berikut ini :




Gambar 3. Ada atribut yang keliru di sebuah entitas

Digambarkan, entitas MATA_KULIAH memiliki atribut-atribut :

KD_MK (Kode Mata Kuliah, sebagai kunci utama/ *primary key*)
NM_MK (Nama Mata Kuliah)
SKS (bobot)
NILAI
NM_DOSEN (Nama Dosen yang Mengajar)

Bila dilakukan hal yang sama dengan pertanyaan di atas, maka dapat diambil kesimpulan bahwa NILAI dan NM_DOSEN bukan merupakan atribut dari MATA_KULIAH.

Bila NILAI dan NM_DOSEN dimasukkan sebagai atribut dari MATA_KULIAH, maka nantinya, siapapun mengambil mata kuliah tersebut akan memiliki NILAI dan nama dosen yang sama. Padahal belum tentu.

Pertanyaan akan menjadi, di mana atribut NILAI harus diletakkan ?. Kalau NM_DOSEN jelas dimasukkan di entitas DOSEN. Sebelum menjawab pertanyaan tersebut, lebih dulu kita amati pembentukan kunci utama (*primary key*). Memang jelas, NPM pasti dipilih menjadi kunci utama dari *file* MAHASISWA, dengan alasan, tidak mungkin ada mahasiswa yang memiliki NPM yang sama. Namun, sesungguhnya, kunci utama diperoleh dari kunci kandidat, dan kunci kandidat diperoleh dari kunci super.

DO NOT COPY

*SUPER KEY*

*Super key* adalah satu atau lebih *field* yang dapat dipilih untuk membedakan (mengkarakteristikkan) antara satu *record* dengan *record* lainnya. Bila *file*nya adalah MAHASISWA, maka satu atau lebih *field* yang dipilih agar dapat membedakan antara satu orang mahasiswa dengan mahasiswa lainnya.

NPM jelas bisa membedakan, NAMA juga bisa, namun dengan syarat tidak ada nama yang sama, gabungan NPM dan NAMA pasti bisa membedakan, apalagi gabungan NPM, NAMA dan TGL_LAHIR. Sehingga, *super key* bisa merupakan kombinasi dari satu atau gabungan *field* yang dapat mencirikan suatu *record*.

*Super key*nya : NPM
NAMA (dengan syarat tidak ada nama yang sama)
ALAMAT (dengan syarat alamat tidak ada yang sama)
TGL_LAHIR (dengan syarat tidak ada tanggal lahir yang sama)
NPM+NAMA
NPM+NAMA+ALAMAT
NPM+TGL_LAHIR
NPM+ALAMAT+TGL_LAHIR
dan berbagai kombinasi lainnya

*CANDIDATE KEY*

Kunci kandidat adalah kunci super dengan jumlah *field* paling sedikit, maka diperoleh : NPM, NAMA, ALAMAT, TGL_LAHIR (karena masing-masing hanya terdiri dari 1 *field* saja).

*PRIMARY KEY*

Kunci utama adalah kunci kandidat yang dipilih dengan kemungkinan kepemilikan nilai data *field* yang berbeda antara satu *record* dengan *record* lainnya. Maka dipilih NPM karena tidak ada mahasiswa yang memiliki NPM yang sama. Jelaslah, kunci utama pastilah merupakan kunci kandidat dan juga kunci super, tetapi sebaliknya, kunci super dan kunci kandidat belum tentu merupakan kunci utama.

*ALTERNATE KEY*

Kunci kandidat yang tidak terpilih menjadi kunci utama disebut dengan kunci alternatif.

Berikut, akan digambarkan di mana atribut (*field*) NILAI dimasukkan ke dalam suatu entitas (*file*) dan apa yang disebut dengan kunci tamu (*foreign key*). Namun, sebelumnya perlu dijelaskan bahwa *file* data dibagi menjadi dua jenis, yaitu *master file* dan *transaction file*. *Master file* sendiri dibagi lagi menjadi 2 jenis yaitu *dynamic master file* dan *reference master file*.

Untuk mempermudah mendapatkan pengertian mengenai apa itu *master file*, salah satunya adalah dengan membayangkan "bila tidak ada (*file*) ini, maka organisasi itu

DO NOT COPY

tidak akan jalan". Misalkan di Perguruan Tinggi, maka *master file*nya bisa terdiri atas MAHASISWA, DOSEN, MATA_KULIAH, UANG_KULIAH, dan sebagainya.

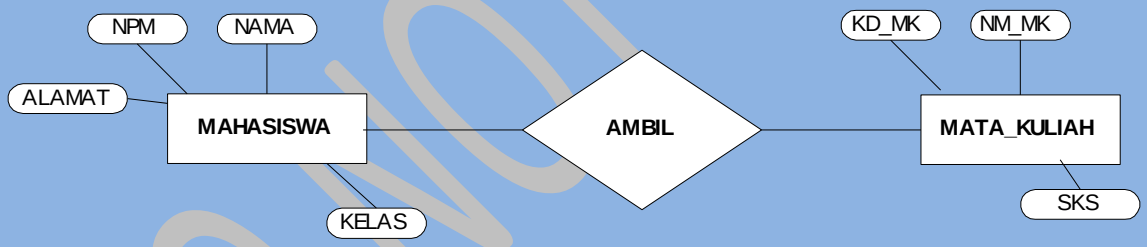
## ***ERD (Entity Relational Diagram)***

ERD Adalah diagram yang menggambarkan hubungan antarentitas (antar*file*). Entitas dilambangkan dengan persegi panjang, relasi (hubungan) digambarkan dengan *diamond*, dan atribut (*field*) dilambangkan dengan elips.

Perhatikan, di kampus. Mahasiswa merupakan sebuah entitas, Mata kuliah juga merupakan entitas. Apa hubungan (relasi) antara mahasiswa dengan mata kuliah ?. Relasinya bisa digambarkan secara sederhana dengan kalimat 'Mahasiswa mengambil mata kuliah.' Apa relasi yang terjadi di sebuah perpustakaan ?, salah satunya adalah 'Anggota meminjam buku.'

Dengan demikian, kata 'relasi' dapat saja diterjemahkan menjadi 'transaksi', misalkan apa transaksi yang terjadi antara mahasiswa dengan mata kuliah ?, atau apa transaksi yang terjadi antara anggota dan buku di perpustakaan ?.

Mahasiswa bertransaksi dengan mata kuliah melalui kata 'ambil'. Maka kata 'ambil' dapat menjadi *file* transaksi, atau kata 'ambil' akan menjadi lambang relasi. Perhatikan gambar berikut ini :



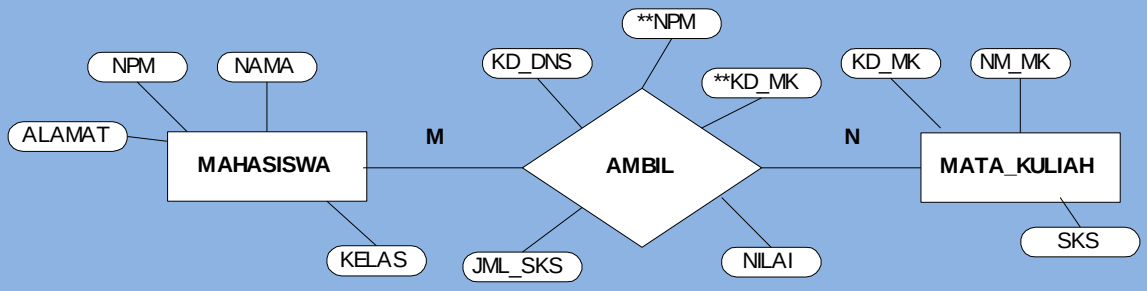

Gambar 4. ERD yang belum lengkap

Jelas, nantinya, di komputer kita akan membuat 3 buah *file data*, yaitu : 2 buah *master file* data, yaitu MAHASISWA dan MATA_KULIAH, dan 1 buah *transaction file*, yaitu AMBIL (nama *file* boleh saja diganti, misalnya *file* AMBIL diganti namanya dengan KRS atau DNS).

Karena AMBIL juga merupakan sebuah *file*, maka dia juga memiliki atribut-atribut. Atribut-atribut apa yang akan kita letakkan di *file* AMBIL ?. Secara sederhana dapat digambarkan dengan kalimat sebagai berikut : 'Apa akibat MAHASISWA (mengambil/ bertransaksi) dengan MATA_KULIAH ?.' Maka, akibatnya adalah akan mendapatkan NILAI, juga akan mengetahui jumlah sks yang diambil, dan sebagainya.

Bila digambarkan secara lebih lengkap, hasilnya :

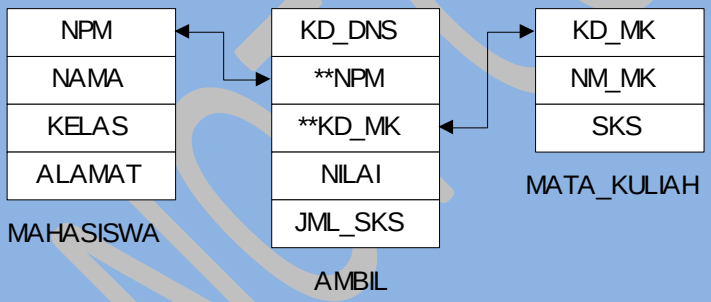
DO NOT COPY




Gambar 5. ERD yang lengkap

**NPM dan **KD_MK adalah kunci tamu, yaitu (yang merupakan) *primary key* dari *file-file* lainnya yang digunakan sebagai penghubung (mendapatkan referensi dari *master-master file* yang saling berelasi*).*

**NPM di *file* AMBIL digunakan untuk mengambil NAMA, ALAMAT dan KELAS dari *file master* MAHASISWA. **KD_MK di *file* AMBIL digunakan untuk mengambil NM_MK dan SKS dari *master file* MATA_KULIAH. Dengan demikian, *file* MAHASISWA dan MATA_KULIAH merupakan *reference master file.*



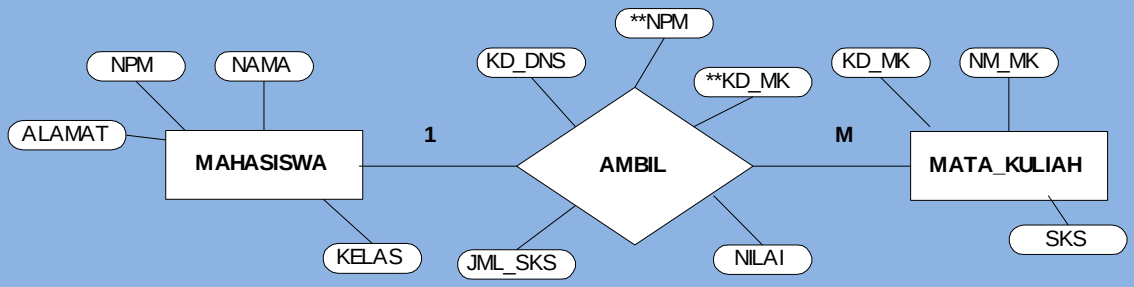

Gambar 6. Skema keterhubungan antar*file*

Sedangkan *dynamic master file* adalah *master file* yang nilai atributnya akan berubah pada saat transaksi dilakukan, misalkan *file* BUKU di perpustakaan, di mana salah satu *field*nya adalah JML_BUKU, maka bila transaksi terjadi (ANGGOTA pinjam BUKU) maka jumlah buku akan berkurang. Atau di suatu *supermarket*, akibat transaksi SUPPLIER memasok BARANG, maka *field* JML_BARANG yang ada di entitas BARANG akan berubah (menjadi bertambah).

Perhatikan derajat kardinalitas di gambar 5. Di sana tertulis M : N (relasinya *many to many*). Bagaimana cara menentukannya ?, ikuti kalimat-kalimat berikut ini :

"Satu **mahasiswa** dapat meng**ambil** satu atau lebih **mata kuliah**" $\rightarrow$ 1 : M

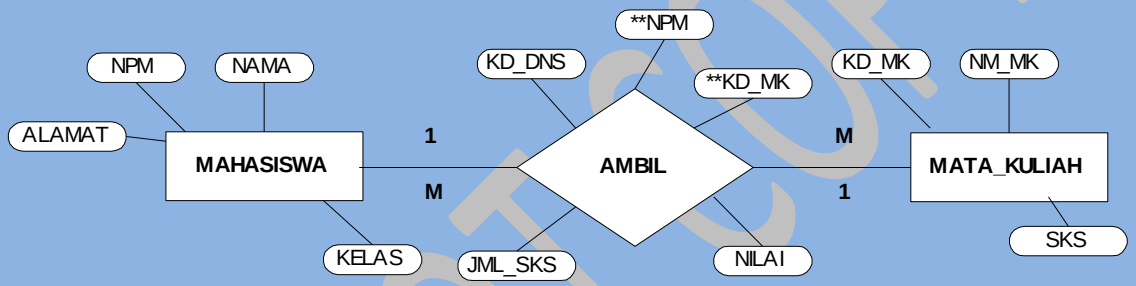
lihat hasilnya di gambar 7 :




Gambar 7. Penentuan *cardinality degree* langkah pertama

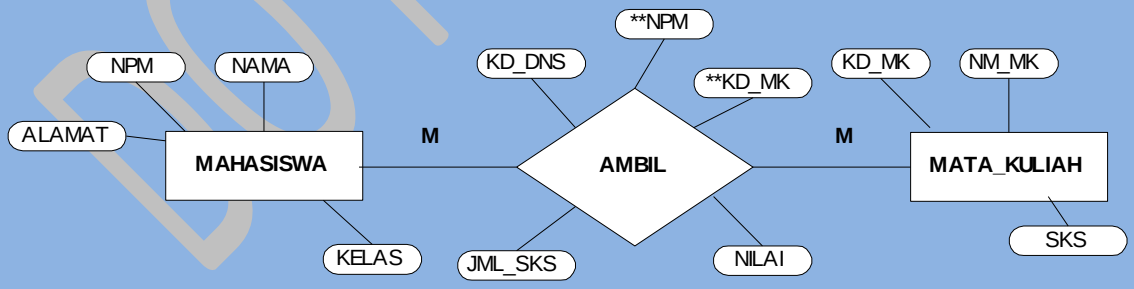
selanjutnya, kalimatnya dibalik :

"Satu **mata kuliah** dapat di**ambil** satu atau lebih **mahasiswa**" → M : 1




Gambar 8. Penentuan *cardinality degree* langkah kedua

Perhatikan gambar 8. Di derajat kardinalitas mahasiswa ada 1 dan M, ambil yang terbesar, yaitu M. Di derajat kardinalitas untuk mata_kuliah, ada M dan 1, ambil yang terbesar, yaitu M. Sehingga derajat kardinalitasnya menjadi M : M (*many to many*).




Gambar 9. Penentuan *cardinality degree* langkah ketiga

Karena dalam matematika, M akan selalu sama dengan M (M = M), maka bila M tersebut diberi nilai 10, apakah pasti : "10 mahasiswa mengambil 10 mata kuliah ?." Karena tidak pasti, maka di salah satunya diubah menjadi N, yang sama juga berarti *many*. Nilai M atau N boleh saja 1, boleh lebih dari satu (banyak/ *many*), suatu saat boleh saja M = N, di lain waktu, boleh saja M tidak sama dengan N. Hasil akhir di gambar 5 di atas.

DO NOT COPY

## MEMASUKKAN DATA KE KOMPUTER

Data dimasukkan ke dalam komputer melalui piranti input, seperti panel kunci ketik (*keyboard*), tetikus (*mouse*), layar sentuh (*touch screen*), dan sebagainya, termasuk dari *file* yang digunakan sebagai input (*input file*).

Sebelum data dimasukkan, maka terlebih dulu kita diminta untuk menyiapkan tempat penampung data (*data store*/ *file*). *File* tersebut harus dibentuk strukturnya terlebih dulu (*create/ creation*). Sebagai contoh, di dalam bahasa pemrograman dBase, langkah tersebut dilakukan dengan menuliskan .CREATE nama_*file*nya. Selanjutnya, kita akan diminta untuk menuliskan nama-nama *field* dan jenis-jenisnya.

## INFORMASI

Secara hafalan, sering dikatakan bahwa informasi adalah hasil atas pengolahan data. Sekarang, kita punya data panjang = 10 dan lebar = 7, luas dari persegi panjang tersebut adalah 70. Apakah 70 itu merupakan informasi ?.

Jawabannya, ya untuk orang tertentu, tetapi bukan bagi orang lainnya. Informasi bersifat individual, sejauh mana angka tersebut memberi makna bagi si penerimanya. Angka 2,70 untuk IPK anda mungkin saja merupakan informasi bagi anda, tetapi bagi saya itu baru merupakan data karena saya ingin mengetahui IPK rata-rata seluruh mahasiswa.

Sehingga, informasi merupakan nilai yang telah memiliki makna bagi seseorang. Sehingga, untuk membuat suatu struktur *field* atau struktur data, seorang *programmer* harus jeli-jeli menyiapkan data agar informasi yang dihasilkan tidak hilang atau tidak bermakna.

Misalkan, ada orang yang menyiapkan data untuk ALAMAT dengan panjang 100 karakter. Ketika ia disodori amplop yang panjangnya 50 karakter, bagaimana ia akan mencetak alamat tersebut ?.

Ada syarat dalam proses normalisasi data tingkat pertama *(first normal form/ 1NF)*, yaitu setiap atribut harus bernilai atomik. Atomik berasal dari kata atom, yaitu benda terkecil yang tidak dapat dibagi-bagi lagi. Sehingga, setiap atribut yang kita buat harus diteliti/ dipikirkan, apakah sudah tidak perlu dibagi-bagi lagi ?.

Sebagai contoh, atribut NAMA. Di Indonesia, mungkin saja atribut tersebut tidak perlu dibagi-bagi lagi karena selamanya nama "Bambang Wahyudi" akan dituliskan seperti itu. Tetapi di Amerika misalkan, nama "Bill Clinton" pada suatu saat akan dituliskan sebagai "Clinton, Bill," bagaimana caranya ?.

Karenanya, di sana, atribut (*field*) NAMA biasanya dibagi minimal atas 2 bagian, yaitu F_NAME (*first name*) dan L_NAME (*last name*), ada pula yang ditambah dengan M_NAME (*middle name*). F_NAME diisi dengan "Bill" dan L_NAME diisi dengan "Clinton" sehingga bila diminta mencetak nama menjadi "Clinton, Bill" cetak dulu L_NAME, baru kemudian F_NAME.

DO NOT COPY

Untuk itu, agar pengolahan data dapat menghasilkan informasi, maka perlu dipersiapkan struktur *file* yang sesuai dengan kebutuhan si pemakai (*user*). Bila anda bekerja di suatu perusahaan, maka cari tahu dulu kebutuhan akan informasi di perusahaan tersebut, sehingga struktur *file* yang anda bentuk tidak perlu diubah-ubah lagi yang justru akan meningkatkan risiko kegagalan proses.

## KEUNTUNGAN dan KERUGIAN BASIS DATA

Bila kita mendengar kata perbandingan (misalkan, keuntungan), maka harus diketahui perbandingannya. Keuntungan basis data adalah keuntungan jika dibandingkan dengan sistem *file* tradisional. Kata tradisional di sini juga jangan dikonotasikan dengan kata "belum memakai komputer."

Pandang sebuah perusahaan, misalkan di sana ada bagian (1) Pembukuan, (2) Produksi, (3) Personalia, dan (4) Pemasaran. Di keempat bagian itu diletakkan masing-masing sebuah komputer PC yang belum dihubungkan satu dengan lainnya, maka, mau tidak mau di setiap komputer tersebut harus memiliki data PEGAWAI.

Akibatnya, keempat komputer tersebut sama-sama memiliki data PEGAWAI yang sebetulnya isinya sama. Ini yang disebut sistem *file* tradisional. Kelemahan utama sistem ini adalah adanya kerangkapan data (*redundancy*), karena data yang sama ditulis berulang-ulang (dalam hal ini sebanyak empat kali).

Kelemahan lainnya adalah *programming oriented*, yaitu format data disesuaikan dengan bahasa pemrograman yang digunakan (belum tentu keempat orang di bagian yang berbeda tersebut memakai bahasa pemrograman yang sama). Data dalam PASCAL tidak dapat digunakan secara langsung oleh data dalam COBOL.

Dapat kita pikirkan lebih dalam, apa lagi kelemahan dalam sistem tradisional ini, misalkan, mungkin saja tidak ada format data yang pasti, misalkan, si A membuat *field* NAMA dengan panjang 20 karakter, si B 30 karakter, si C 25 karakter, dan si D 40 karakter. Si A menulis nama "Bambang Wahyudi", si B menulis nama itu dengan "B. Wahyudi", si C menulisnya dengan "Bambang W.", dan si D menulis "BAMBANG WAHYUDI", tidak ada keseragaman.

Misalkan lagi, bila pegawai "Ali" pindah rumah, ia harus melaporkan kepindahannya ke empat bagian yang berbeda, kalau tidak, maka akan timbul masalah, data alamat mana yang benar untuk si "Ali" ?.

Begitu juga ketika Pimpinan meminta atau ingin mendapatkan laporan kemajuan perusahaannya, maka ia harus meminta laporan dari seluruh bagian yang ada. Ia tidak dapat mengakses langsung di komputer (karena belum ada integrasi data).

Secara kejadian se hari-hari di Bank, bila sebuah bank belum menerapkan sistem basis data, maka kita tidak dapat menabung di cabang tertentu dan mengambil di cabang

lainnya (sering diistilahkan dengan sistem *online* untuk menggambarkan penggunaan basis data).

Kelemahan-kelemahan yang ada di sistem *file* tradisional itu ditanggulangi oleh sistem basis data. Konsep utama basis data adalah meniadakan kerangkapan data, dengan menempatkan data di satu media penyimpanan tertentu (misalkan pada *hard disk* di dalam *server*) dan setiap komputer dihubungkan oleh sebuah jaringan menuju *server* tersebut. Semua *user* yang memerlukan data mengakses pada data yang sama.

Karena sulit dan rumitnya mendisain sebuah basis data, maka kelemahan basis data adalah diperlukannya tenaga ahli, misalkan orang yang berkedudukan sebagai *Data Base Administrator* (DBA). DBA bertugas untuk mendisain rancangan basis data, memberi hak dan wewenang kepada setiap *user*, menentukan bentuk-bentuk akses data, dan sebagainya.

Hal yang lebih fatal dapat terjadi yang merupakan kerugian basis data adalah jika data rusak, maka seluruh *user* akan terganggu, sedangkan pada sistem *file* tradisional, bila data si A rusak, maka si B, C, dan D masih dapat terus bekerja.

Banyak *tool* (alat) yang digunakan untuk merancang suatu sistem kerja, misalkan saja *Data Flow Diagram/* DFD (Diagram yang menggambarkan aliran data dari suatu sumber menuju ke tujuan). Lambang-lambang DFD terdiri atas (1) Terminator, disebut juga kesatuan luar, disebut juga sumber atau tujuan. Terminator dilambangkan dengan persegi panjang. Terminator menunjukkan data yang akan diolah oleh sistem berasal dari (kesatuan) mana, dan akan ditujukan (dilaporkan) kepada (kesatuan) mana. Kesatuan tersebut dapat merupakan jabatan seseorang, sebuah organisasi, dan sebagainya. (2) alur data, berupa arah panah yang membawa jenis data, (3) proses, berupa lingkaran yang berisi suatu kegiatan kerja, dan (4) penyimpan data (*data store*), berupa garis paralel yang merupakan nama *file* yang digunakan sebagai tempat acuan berkas untuk atau hasil suatu proses.

DFD terdiri dari diagram konteks, yaitu gambaran umum (secara garis besar) sistem yang akan dibuat. Selanjutnya dijabarkan menjadi diagram nol (*zero*), dan dijabarkan lagi menjadi *level-level* yang lebih detil lagi.

Dari gambaran DFD tersebut akan diperoleh *file-file* yang akan digunakan (*data store*). *File-file* tersebut akan saling direlasikan di ERD (telah dibahas di atas).

Alat lain yang digunakan adalah kamus data (yang menjabarkan makna dan ketentuan dari setiap atribut yang digunakan), misalkan : NPM, jenis karakter, panjang 8, karakter pertama berisi angka 1, 2, 3, 4, 5, atau 6, di mana misalkan angka "1" untuk fakultas ilmu komputer, jenjang strata satu, jurusan sistem informasi, dan seterusnya.

Dan banyak lagi alat yang dapat digunakan yang kesemuanya digunakan untuk menyempurnakan sistem yang akan dibangun agar terhindar dari kesalahan atau kehilangan informasi yang dibutuhkan.

DO NOT COPY

## MEDIA PENYIMPANAN DATA

Memori utama komputer (*main memory* atau *main storage* atau *primary storage*) dapat diumpamakan STM pada manusia, sehingga hanya bersifat amat sementara (ingatan hilang bila listrik dipadamkan/ bersifat volatil). Untuk menjadikannya LTM, maka data yang akan diingat direkam ke media penyimpanan sekunder (*secondary memory*). Banyak alat yang ditawarkan untuk menyimpan data digital, seperti *hard disk,* disket, *magnetic tape, compac disc,* dan sebagainya.

Secara umum, media penyimpanan sekunder dibagi atas 2 jenis, yaitu (1*) Serial (sequential) access storage device (SASD),* dan (2) *Direct access storage device (DASD).* SASD memiliki prinsip kerja seperti sebuah kaset lagu, yaitu jika kita akan merekam atau mendengarkan lagu, maka lagu kedua akan didahului lagu pertama, dan seterusnya. DASD memiliki prinsip kerja seperti sebuah CD lagu, kita tidak perlu menyetel lagu pertama jika ingin mendengarkan lagu ke dua.

Penyimpanan data di dalam sebuah media penyimpanan tidak dilakukan sembarangan karena suatu saat data yang telah disimpan itu akan diambil kembali (*retrieve*) dan akan diolah. Karenanya, dalam proses penyimpanan data dikenal pula istilah alamat (*address*). Pemakai kini tidak perlu pusing-pusing lagi memikirkan di mana alamat sebuah data di simpan di suatu media penyimpanan, semua sudah diatur oleh sistem operasi yang ada di setiap komputer. Namun demikian, pada teknik pengalamatan mutlak, pemakai boleh mendeklarasikan sendiri di alamat mana data tersebut akan di simpan. Sulitnya, bila ada seribu data yang sudah dimasukkan, maka ia harus tahu di mana alamat suatu data ketika ia akan mengolahnya.

Data yang disimpan di suatu media penyimpanan juga perlu diorganisasikan agar sesuai dengan teknik atau cara pengolahan data yang akan dilakukannya kemudian. Ada 4 teknik dasar pengorganisasian data, yaitu (1) *Sequential,* (2) *Relative,* (3) *Index Sequential*, dan (4) *Multi key.*

Andaikan kita memiliki sekumpulan lagu di suatu media penyimpanan dan kita mau mendengarkannya, maka teknik *sequential* seperti kita menyetel lagu lewat kaset, *relative*, seperti menggunakan CD, *index sequential* seperti MP3 (mencari sebuah kata dalam kamus), dan *multi key* kita dapat memilih lagu baik berdasarkan judul, atau nomor urut, atau nama penyanyinya (beberapa kemungkinan yang dapat memilih lagu secara langsung pada lagu yang ingin didengarkan, medianya tidak dapat dicontohkan).

### AKSES DATA UNTUK MENGHASILKAN INFORMASI

Akses (pengolahan data) sangat berkaitan dengan teknik pengorganisasian data yang telah dilakukan sebelumnya. Bila kita mengorganisasikan data secara *sequential,* maka mau tidak mau kita mengaksesnya juga secara *sequential*. Tetapi, bila kita mengorganisasikannya secara *relative, index sequential,* maupun *multi key*, selain kita dapat mengaksesnya secara *direct*, kita juga dapat melakukan akses secara *sequential*.

Jadi, pada dasarnya, akses data terdiri atas 2 cara, yaitu (1) *direct access*, dan (2) *sequential access*. Meskipun kita memakai CD, kita tetap dapat mendengarkan lagu

secara berurut (*sequential*), sebaliknya, jika kita menyetel lagu lewat kaset, maka tidak mungkin kita dapat mendengarkan lagu ke 5 tanpa melakukan *forward* atau *rewind*.

Ada 2 model penggunaan akses data, yaitu (1) *batch*, dan (2) *iterative.* Model *batch* adalah pengaksesan data yang dilakukan secara berkelompok atau *group*, misalkan dalam kehidupan sehari-hari pada proses penarikan undian, semua kupon undian dikumpulkan dulu baru kemudian diproses untuk diambil pemenangnya. Model *iterative* adalah proses yang dilakukan secara langsung, seperti misalkan pengisian KRS. Mahasiswa yang telah selesai mengisi, KRSnya akan dicetak langsung tanpa menunggu semua mahasiswa selesai mengisi KRSnya.

Secara umum, hal-hal yang dilakukan terhadap sebuah *file* (berkas) adalah : (1) *creation* (menyiapkan struktur *file*nya), (2) *update*, seperti *inserting/ adding* (penambahan data), *modification* (perubahan data), *deletion* (menghapus data), (3) *retrieval* (peraihan kembali data) baik untuk proses (*inquiry*) maupun pembuatan laporan data (*report generation*), (4) *maintenance* (perawatan) seperti *restructuring* (perubahan struktur *file*), dan *reorganization* (perubahan organisasi data).

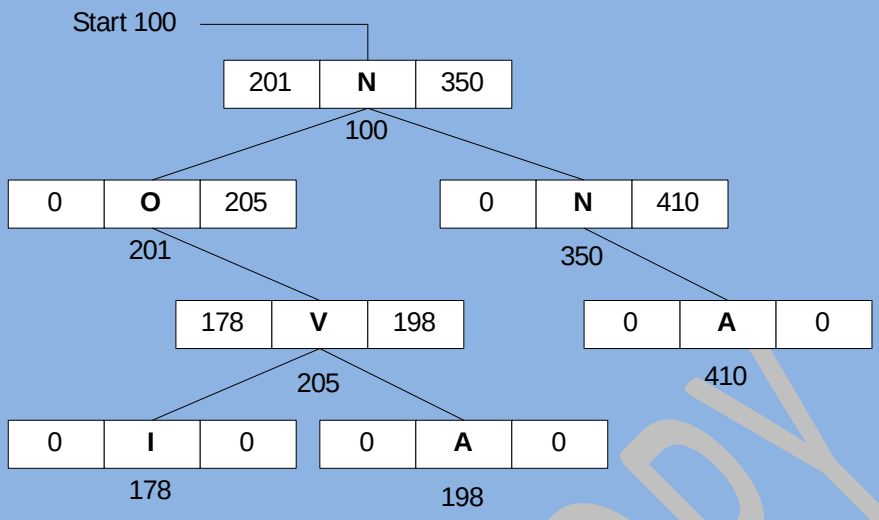
Di komputer, data juga mengalami perpindahan (transportasi). Bagaimana data dimasukkan melalui *keyboard*, ditampilkan ke monitor, selanjutnya melalui *main memory*, melalui rangkaian-rangkaian elektronik lainnya hingga disimpan di media penyimpanan. Begitu juga ketika data di-*retrieve* dari media penyimpanan, diproses di *main memory*, ditampilkan ke monitor atau dicetak ke *printer* atau dikirim melalui jaringan ke komputer lain, dan sebagainya.

Transportasi data dari alamatnya masing-masing ke suatu rangkaian elektronik tertentu dan sebaliknya melalui bus-bus data akan dipandu/ dikoordinasikan oleh sebuah *software* yang disebut dengan *operating system*. Contoh sistem operasi yang sudah banyak dikenal misalkan *Disk Operating System* (DOS), *Windows*, UNIX, dan sebagainya. Sistem operasi juga berfungsi untuk mengaktifkan piranti-piranti komputer. Untuk mengetahui bagaimana rumitnya algoritma (hanya) untuk menampilkan data yang sedang diketik ke layar monitor, cobalah membuatnya dengan bahasa mesin/ bahasa rakitan (*assembler*).

## ***BINARY TREE***

Para perancang sistem operasi selalu berusaha memikirkan algoritma yang tepat, efektif dan efisien guna "menjaga" data yang ada agar tidak hilang. Seperti ketika suatu data akan disimpan di suatu media penyimpanan, bagaimana menentukan alamatnya agar dapat diraih kembali, bagaimana agar keterhubungan antara satu bagian data dengan bagian data lain tidak terputus, dan sebagainya.
Salah satu metoda yang digunakan adalah *binary tree*. Alamat dan keterkaitan antardata dapat digambarkan sebagai pohon biner, seperti contoh berikut ini :

DO NOT COPY




Gambar 10. Contoh keterkaitan antardata

*Start* menunjuk ke alamat 100, yaitu "N". "N" sendiri menunjuk alamat kiri ke 201 (yaitu "O"), dan kanan ke 350 (yaitu "N"). Buatlah algoritma agar informasi yang akan dihasilkan adalah "NOVIANA".

Dengan berbagai pertimbangan, tidak semua perancang komputer membuat algoritma untuk melakukan perhitungan data matematis seperti yang dilakukan oleh manusia pada umumnya (secara *infix)*. Ada komputer yang mengolahnya secara *prefix* dan ada yang secara *postfix.*

Contoh bentuk *infix* : A + B
*prefix* : + A B
*postfix* : A B +

Lambang yang digunakan manusia untuk melakukan penjumlahan angka 5 dan 7, adalah 5 + 7 (*infix*). 5 dan 7 disebut *operand*, dan + disebut *operator*. Untuk komputer yang melakukan proses matematis secara *prefix*, kalimatnya diubah dengan "tambahkan 5 dengan 7" atau dilambangkan dengan + 5 7, dan untuk *postfix*, kalimatnya adalah "5 dan 7 ditambahkan", atau dilambangkan dengan 5 7 +.

Bagaimana penulisan notasi *infix* : A + B * C + D secara *prefix* dan *postfix* ?.

*Prefix* :
langkah 1, tentukan mana yang pertama kali yang akan diproses, diperoleh B * C. Misal B * C diganti dengan E, maka soalnya menjadi : A + E + D

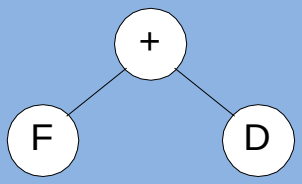
$$A + \underline{B * C} + D$$
$$E$$

langkah 2, tentukan kembali mana yang akan diproses pertama kali, diperoleh A + E. Bila A + E diganti dengan F, maka soalnya menjadi F + D

$$\underline{A + E} + D$$
$$F$$

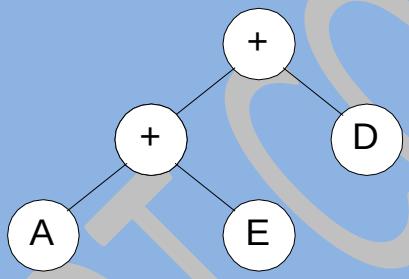
DO NOT COPY

langkah 3, karena sudah menjadi 2 *operand*, maka notasi *prefix*nya menjadi + F D



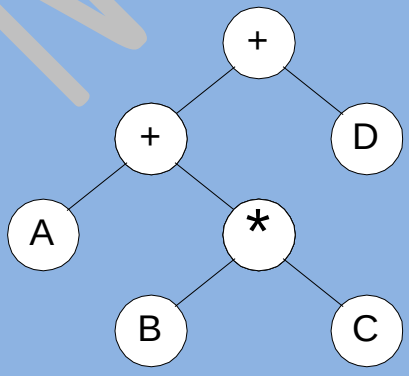

Gambar 11a. *Binary tree* dari F + D

langkah 4, kembalikan nilai F sebenarnya. F = A + E, *prefix*nya = + A E, sehingga notasi keseluruhannya menjadi + + A E D




Gambar 11b. *Binary tree* dari (A + E) + D

langkah 5, kembalikan nilai E sebenarnya. E = B * C, *prefix*nya = * B C, sehingga notasi keseluruhannya menjadi **+ + A * B C D**




Gambar 11c. *Binary tree* dari A + (B * C) + D

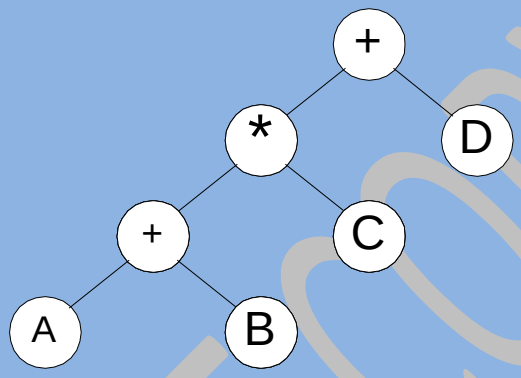
Ekspresi matematis A + ( B * C ) + D = A + B * C + D.

*Postfix* :
langkah 1, tentukan mana yang pertama kali yang akan diproses, diperoleh B * C. Misal B * C diganti dengan E, maka soalnya menjadi : A + E + D

DO NOT COPY

langkah 2, tentukan kembali mana yang akan diproses pertama kali, diperoleh A + E. Bila A + E diganti dengan F, maka soalnya menjadi F + D
langkah 3, karena sudah menjadi 2 *operand*, maka notasi *postfix*nya menjadi F D +
langkah 4, kembalikan nilai F sebenarnya. F = A + E, *postfix*nya = A E +, sehingga notasi keseluruhannya menjadi A E + D +
langkah 5, kembalikan nilai E sebenarnya. E = B * C, *postfix*nya = B C *, sehingga notasi keseluruhannya menjadi **A B C * + D +**

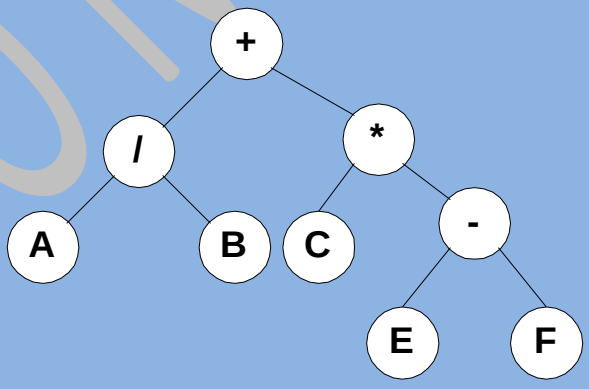
Untuk soal : (A + B) * C + D bentuk *binary tree*nya sebagai berikut :




Gambar 12. Skema *binary tree* untuk (A+B)*C+D

*Prefix*nya : + * + A B C D
*Postfix*nya : A B + C * D +

Dari gambar *binary tree*, kita dapat secara langsung menentukan bentuk notasi *infix, prefix,* maupun *postfix*nya. Perhatikan gambar berikut ini.




Gambar 13. Tentukan *infix, prefix*, dan *postfix*nya

*Infix* : Ingat, konsep *binary tree* adalah "pembelahan dua." Jika gambar 13 itu dibelah menjadi dua, maka belahan kiri (*left substree*) terdiri dari A / B, dan belahan kanan (*right substree*) adalah C * - E F, sementara yang di tengah (*root*) adalah +. Di belahan kiri kita sudah mendapatkan notasi (A / B). Di belahan kanan, kita belah lagi menjadi belahan kiri2 C, dan belahan kanan2 E - F, sedangkan yang di tengah (*root*) adalah *.

DO NOT COPY

Hasil pembelahan itu kini kita gabungkan (setiap operator atau *root*), kita letakkan di tengah. Notasinya menjadi : (A + B) + ( C * (E - F))
kiri kanan
Hilangkan tanda kurung pemisah yang tidak penting (bila dibuang, hasil pemrosesan tidak berubah). Notasinya menjadi : A + B + C * (E - F)

*Prefix :* Karena *operand* berada di depan, sehingga untuk notasi A + B dituliskan dengan + A B, maka bila bila kita sebut A adalah kiri, + adalah tengah, dan B adalah kanan, maka rumusnya adalah tengah-kiri-kanan secara rekursif (lakukan berulang di setiap *node* mana yang sedang dikunjungi)

Lakukan rumus itu dari atas, kita dapat tengah + (sementara hasilnya adalah +).
Setelah tengah, kita menuju ke kiri, yaitu /. Stop di situ, kita ulangi lagi rumus semula, tengah-kiri-kanan. Tengah = /, kiri = A dan kanan = B. Hasil berikutnya menjadi + /.

Dari / kita menuju ke kiri lagi, kita dapat A. Stop di situ, ulangi lagi rumus tengah-kiri-kanan. Karena tidak ada lagi yang lebih kiri dari A, maka A dijadikan hasil, sehingga hasilnya menjadi + / A. Setelah tidak ada yang di kiri lagi, maka lanjutkan rumus, setelah kiri adalah kanan, yaitu B. Ulangi lagi rumus tengah-kiri-kanan, karena tidak ada lagi yang berada di kiri maupun di kanan B, maka jadikan B sebagai hasil, sehingga hasilnya menjadi + / A B.

*Substree* kiri sudah selesai, maka lanjutkan ke *substree* kanan, lakukan sama dengan rumus tengah-kiri-kanan. Maka hasil akhirnya akan diperoleh : + / A B * C - E F.

*Postfix* : Karena operator di belakang *operand*, maka rumusnya : kiri-kanan-tengah.
(Ulangi seperti yang telah diuraikan di penyelesaian notasi *prefix* dengan rumus yang berbeda).

Dimulai dengan *substree* kiri, kita peroleh : A B /.
Dilanjutkan dengan *substree* kanan, kita peroleh : C E F - * +.
Hasil akhirnya : A B / C E F - * +

*Prefix* menempatkan puncak *(*akar/ *root*) di awal notasi, sedangkan *postfix* menempatkan puncak (*root*) di akhir notasi. Dengan sering melakukan latihan, dengan hanya melihat gambar *binary tree*nya, kita akan mudah menemukan pola pergerakan untuk menentukan notasi *prefix* dan *postfix*nya.

Bisakah anda mencari jalannya bila diketahui *postfix* = A B C - + D E F / + *, maka *prefix*nya = * + A - B C + D / E F, dan *infix*nya = (A + B - C ) * ( D + E / F) ?. Gambarlah *binary tree*nya.

## SISTEM INFORMASI MANAJEMEN dan SISTEM PENUNJANG KEPUTUSAN

Sistem adalah sekumpulan elemen yang masing-masing memiliki fungsi masing-masing dan secara bersama-sama mencapai tujuan sistem itu. Mobil merupakan sebuah sistem,

DO NOT COPY

karena di dalamnya banyak elemen yang memiliki fungsi masing-masing, seperti setir untuk kendali, rem untuk memberhentikan, gas untuk menjalankan, radiator untuk pendingin, dan sebagainya, yang secara bersama-sama mencapai tujuan dari mobil yaitu sebagai alat transportasi.

Begitu juga di perusahaan, ada bagian pemasaran, ada bagian produksi, ada bagian pembukuan, dan sebagainya yang kesemuanya bekerja untuk mencapai tujuan perusahaan itu, misalkan mendapatkan keuntungan finansial. Di setiap bagian di perusahaan tentu memiliki data dan informasi. Agar bermanfaat, data dan informasi tersebut dikelola (manajemen) guna menopang kebutuhan para manajer dalam mengendalikan perusahaannya.

Manajer tingkat atas (*top management*) memiliki jenis keputusan yang bersifat strategis, manajer tingkat menengah (*middle management*) memiliki keputusan yang bersifat taktis, dan manajer tingkat bawah (*lower management*) memiliki keputusan yang bersifat operasional, semuanya membutuhkan data dan informasi.

Informasi yang baik adalah informasi yang diberikan sesuai dengan kebutuhannya, baik pada kelengkapan materinya, waktu pemberian informasinya, keakuratan datanya, dan sebagainya. Misalkan saja, manajer pemasaran membutuhkan informasi mengenai kondisi pasar, kondisi pesaing, kondisi ekonomi makro, kekuatan perusahaan, kemampuan finansial perusahaan, dan sebagainya.

Agar informasi dapat dilakukan secara cepat dan akurat, maka pada masa kini, tak ada pilihan lain selain memanfaatkan komputer yang di dalamnya dibentuk sistem basis data. SIM adalah kerjasama yang harmonis antara manusia dan mesin (komputer). Sedapat mungkin semua alat-alat kantor dibuat berangkaian dengan komputer (*office automation*), misalkan pemanfaatan *e-mail, tele- conference, e-voice, internet, facs,* dan sebagainya.

Ketersediaan informasi dan kecepatan mendapatkan informasi merupakan "senjata" yang ampuh dalam memenangkan persaingan global dewasa ini. Dengan semakin sibuknya para manajer, jadwal kegiatan begitu ketat, menjadikan mereka tidak sempat lagi membaca laporan yang bertumpuk dari setiap bagian di perusahaannya. Karenanya diperlukan suatu cara yang kini disebut dengan *decision support system,* (DSS) yaitu sistem komputer yang dapat membantu para manajer untuk mengambil keputusan yang relatif tepat dan cepat.

Para manajer (*user*) tidak perlu lagi membaca laporan yang bertumpuk, tetapi cukup tinggal meng-"klik" saja, maka data dan informasi muncul di komputernya dalam sekejap. Data dan informasi juga bukan lagi sekadar teks melainkan berupa grafik, animasi dan suara yang akan lebih cepat dicernanya.

Namun demikian, tidak semua manajer "pandai" memakai komputer atau membaca informasi yang diberikan oleh komputer, karena mungkin saja latar belakang pendidikan mereka tidak memungkinkan. Karenanya manajer memerlukan seorang perantara (intermediator) untuk mengoperasikan komputer atau "menterjemahkan" laporan komputer.

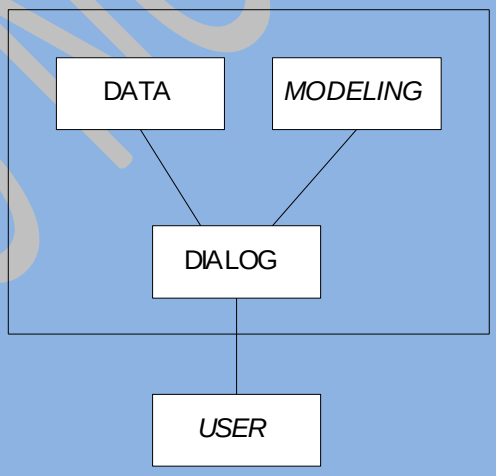
Orang yang bertugas merancang konfigurasi DSS disebut *DSS builder* atau fasilitator antara kebutuhan informasi yang akan dihasilkan dengan sistem teknologi yang digunakannya. Ia harus memiliki kapabilitas dan menguasai masalah (mengetahui kebutuhan informasi) sehingga interaksi antara *user* dan sistem dapat dijalankan dengan baik.

*DSS builder* dibantu oleh *technical supporter* yang mensuport kebutuhan data baru atau model analisis baru, atau apa saja yang diperlukan sistem untuk menunjang kebutuhan akan informasi dari *user*.

*Technical supporter* dibantu lagi oleh *toolsmith*, yang bertugas untuk mengembangkan teknologi baru, bahasa pemrograman baru, *hardware* dan *software* baru, meningkatkan efisiensi dan keterkaitan di antara sub-sub sistem di dalamnya.

Dalam DSS, data dikumpulkan ke dalam sebuah *database* dan diorganisasikan oleh *database management software* (DBMS), dan model penyampaian informasinya dikumpulkan ke dalam *model base* dan diorganisasikan oleh *model base management software* (MBMS). Keduanya diorganisasi oleh *dialogue generation and management software* (DGMS) yang bermanfaat sebagai *interface* antara *user* dan sistem.

Data di dalam *database* mencakup : sumber data eksternal perusahaan dan data di dalam perusahaan (keuangan, pemasaran, personalia, produksi, dan sebagainya). *Model base* yang ada disesuaikan pula oleh tingkatan manajer dan jenis-jenis pengambilan keputusannya, maka ada *strategic models*, *tactical models*, dan *operational models*. Skema sederhananya dapat digambarkan sebagai berikut :




Gambar 14. Paradigma *Dialog-Data-Model* dalam DSS

Komputer tersebut juga didisain untuk dapat memberikan "nasihat" atau "pilihan alternatif" keputusan yang harus diambil segera oleh pimpinan. Karenanya, komputer itu harus mencakup pula *expert system*, yaitu suatu *software* yang dapat membantu para manajer mengikuti cara berpikir para ahli di bidangnya.

DO NOT COPY

Kita pernah mendengar super komputer "*Deep Blue*", yaitu komputer yang dirancang untuk dapat bermain catur dan pernah mengalahkan para pecatur tingkat dunia. Komputer itu menggunakan konsep sistem pakar (*expert system*), karena di dalamnya dimasukkan pemikiran para ahli pemain catur, sehingga bagaimana langkah selanjutnya bila lawan menggerakkan sebuah bidak, akan diproses seperti bagaimana seorang ahli catur memikirkan hal itu dalam melakukan pergerakan berikutnya.

Tentunya, sistem pakar untuk komputer di perusahaan bisa saja merupakan penganalisaan masalah keuangan, tingkat persaingan, sistem penggajian, dan sebagainya. Ada hal yang lebih canggih lagi yaitu komputer yang dapat menganalisa kesalahan-kesalahan yang telah dilakukannya, dengan menambahkan konsep *artificial inteligence* (AI).

Kecerdasan buatan yang "ditanamkan" itu akan mampu menganalisa kesalahan-kesalahan yang pernah dilakukannya, sehingga kesalahan serupa akan diusahakan dihindarinya. AI berkaitan erat dengan *paralel system*, sebagaimana manusia dapat mengakses data melalui berbagai jalur susunan saraf (tidak serial). Pada hal yang sama, komputer juga dilatih melalui proses pembelajaran, misalkan bagaimana ia dapat mengenali benda tertentu, misalkan "pensil". Komputer, melalui piranti sensornya diberikan benda "pensil" dan direkam di memorinya bahwa benda tersebut bernama "pensil". Hal tersebut dilakukan berulang-ulang, karena bila hanya sekali saja, mungkin komputer "lupa" akan nama benda tersebut. Bila hanya digunakan algoritma saja, bagaimana anda dapat mendefinisikan sebuah "pensil" ?, baik dari kegunaannya, bentuknya, sifatnya, dan sebagainya. Bisakah definisi itu memberi nilai pasti yang tidak akan keliru dengan "*ballpen*" atau "kuas" ?.

## KESIMPULAN

Diharapkan, bila konsep mengenai data dan informasi ini dapat dicerna, maka setiap mahasiswa tinggal mengembangkan diri dengan membaca buku-buku paket atau buku-buku lain. Dengan landasan yang kuat, maka pengembangan akan menjadi mudah. Pesan : jangan hanya pandai menghafal karena akan mudah hilang bila berhadapan dengan penguji, untuk itu, pandailah mendapatkan pengertiannya agar satu pertanyaan dapat dijawab dari berbagai jalur pengertian yang dimilikinya. Semoga berhasil.

Contoh-contoh pertanyaan :

## SISTEM BASIS DATA

1. Jelaskan apa itu "basis data."
2. Konsep utama basis data adalah *non redundancy data*, jelaskan.
3. Jelaskan mengapa basis data memerlukan *memory space* yang lebih besar ketimbang sistem *file* tradisional ?
4. Tunjukkan salah satu keterkaitan antar *file* dalam suatu basis data di perpustakaan
5. Mengapa dalam basis data diperlukan tenaga ahli seperti *data base administrator* ?
6. Sebutkan dan beri contoh normalisasi data tingkat 1 sampai 3
7. Apa saja kerugian basis data ketimbang sistem *file* tradisional, dan sebaliknya ?
8. Bagaimana cara saudara menentukan sebuah *key field* dalam sebuah *file* ?

9. Sebutkan dan beri contoh perintah-perintah *query* yang berhubungan dengan *Data Manipulation Language* (DML) dan *Data Definition Language* (DDL).
10. Jelaskan apa itu *distributed data base.*

## ANALISIS dan PERANCANGAN SISTEM

1. Apa saja analisis yang anda lakukan pada waktu merancang sebuah sistem komputerisasi, dan pada waktu pengembangan sistem komputerisasi.
2. Siapa saja (jabatan) orang-orang yang terlibat pada perancangan sistem komputerisasi.
3. *Tools* atau alat-alat apa saja yang digunakan untuk membantu membuat perancangan sistem komputerisasi, dan jelaskan fungsinya masing-masing.
4. Jelaskan tahapan-tahapan *System Life Cycle* atau *System Development Life Cycle.*
5. Sebutkan keterhubungan antara *Data Flow Diagram* dengan *Entity Relational Diagram.*
6. Sebutkan siapa saja *user* dari sistem komputerisasi, apa saja peran mereka dalam membantu seorang *system analyst* dalam membuat perancangan sistem komputerisasi.
7. Bagaimana anda dapat menelusuri kesalahan bila sistem komputerisasi yang anda buat ternyata memiliki kesalahan (kecacatan) ?
8. Siapa saja yang anda ajak untuk mencoba sistem komputerisasi yang baru selesai anda buat, dan apa peran mereka dalam uji coba tersebut ?
9. Bagaimana cara anda bila sistem komputerisasi lama yang sedang berjalan akan anda ganti dengan sistem komputerisasi yang baru selesai anda buat ?
10. Dokumentasi apa saja yang anda lakukan selama pembuatan sebuah sistem komputerisasi dan apa gunanya dokumentasi tersebut ?

## BERKAS dan AKSES

1. Sebutkan jenis-jenis berkas dan jenis-jenis akses di komputer.
2. Sebutkan keuntungan penggunaan berkas *sequential* dibanding dengan relatif, dan sebaliknya.
3. Sebutkan contoh pengolahan data secara *batch.*
4. Sebutkan macam-macam teknik pengalamatan.
5. Sebutkan contoh pengaksesan data yang menggunakan *multi key*.
6. Jelaskan apa yang dimaksud dengan *buffer* dan apa gunanya ?.
7. Sebutkan macam-macam media penyimpanan data yang termasuk pada *Direct Access System Device* (DASD) dan *Sequential Access System Device* (SASD) ?
8. Apa yang dimaksud dengan *collision* pada teknik pengalamatan dan bagaimana pemecahannya bila itu terjadi.
9. Sebutkan keuntungan dan kelemahan *main storage* dibanding dengan *auxuliary storage (secondary storage)* dalam penyimpanan data.
10. Jelaskan beda antara *restructuring* dan *reorganization* dalam sebuah berkas.

DO NOT COPY

## STRUKTUR DATA

1. Apa konsep dari "isi" mata kuliah "struktur data" ?.
2. Sebutkan jenis-jenis operasi yang terdapat pada *stack* dan *queue* dan berikan contoh-contohnya.
3. Apa beda struktur data linier dan non liner, berikan contohnya.
4. Apa beda struktur data sederhana "*Array*" dan "*Record*", jelaskan dalam penggunaannya.
5. Jelaskan tiga macam teknik sortir data yang anda ketahui dan jelaskan perbedaan konsepnya.
6. Apa yang dimaksud dengan "antrean berprioritas" pada *queue* ?
7. Gambar sebuah pohon biner untuk "A + B - C * D", serta tentukan notasi *prefix* dan *postfix*nya.
8. Bagaimana menambahkan sebuah data di tengah *linked-list* ?
9. Kapan saja operasi ISEMPTY(S) bernilai "*True*" pada sebuah *stack*.
10. Jelaskan beda antara "*grounded header list*" dengan "*circular header list*" pada sebuah *linked-list*.

## SISTEM INFORMASI MANAJEMEN (SIM)

1. Apa pendapat anda mengenai mengapa informasi perlu dikelola ?
2. Ada pendapat bahwa SIM merupakan bagian dari CBIS (*computer base information system*), tetapi ada yang berpendapat sebaliknya. Bagaimana anda menjelaskan fenomena ini ?
3. Informasi apa saja yang mungkin diperlukan untuk bagian pemasaran di suatu perusahaan ?
4. Apa saja yang dapat dilakukan agar *office automation* dapat menopang kinerja SIM dalam mengelola informasi ?
5. *Executive Information System (EIS)* merupakan bagian dari SIM. Sebutkan data apa saja yang dibutuhkan untuk menghasilkan informasi bagi para eksekutif ?
6. Apa manfaat dimasukkannya unsur *Expert System* (ES) dalam menopang kinerja SIM ?
7. Jelaskan hubungan antara basis data dengan SIM.
8. Jelaskan apa yang dimaksud dengan keputusan yang bersifat strategis, taktis dan operasional, berikan contoh-contohnya dalam dunia nyata.
9. Sebutkan syarat-syarat bahwa informasi (laporan) dapat dikatakan "baik".
10. Sebutkan bagian-bagian yang termasuk dalam CBIS.

## SISTEM PENUNJANG KEPUTUSAN (SPK)

1. Sebutkan 3 tingkatan dari SPK, dan 5 jenis jabatan orang yang masuk di dalamnya.
2. Pertimbangan apa saja yang harus dilakukan bila perusahaan akan menerapkan SPK ?
3. Apa yang dimaksud dengan SPK spesifik (*Specific Decision Support System*) ?
4. Apa kriteria yang perlu diambil untuk menentukan pemilihan terhadap *software* dan *hardware* SPK ?
5. GDSS *(group decision support systems*) merupakan bagian dari SPK. Jelaskan manfaat dari GDSS dalam menopang keberhasilan sebuah SPK.

DO NOT COPY

6. Jelaskan manfaat *expert system* dalam sebuah SPK.
7. Perkembangan *hardware* dan *software* komputer yang demikian pesat dapat membuat kesan SPK yang sudah dibuat menjadi ketinggalan jaman. Sebutkan alasan-alasan yang perlu dikemukakan agar *maintenance* terhadap DSS *generator*nya dapat diterima pimpinan perusahaan.
8. Jika SPK sudah mencakup sistem pakar di dalamnya, perlukah seorang pengambil keputusan mengadakan rapat dengan dewan direksi atau para karyawan untuk menetapkan keputusannya ?. Jelaskan.
9. Sebutkan kriteria anggota dari tim pengembang *Executif Information System* (*Executif Support System*) agar sistem yang dibuatnya bermanfaat bagi para eksekutif.
10. Bagaimana format yang cocok untuk menampilkan (mempresentasikan) laporan atau data di monitor para eksekutif agar bersifat efektif ?

## ***TESTING* dan IMPLEMENTASI SISTEM**

1. Sebutkan jenis-jenis kesalahan dari sebuah sistem komputerisasi, berikan contoh-contohnya.
2. Pada tahapan ke berapa testing dan implementasi sistem dalam siklus kehidupan sistem (*system life cycle*).
3. Siapa saja yang berhak melakukan pengujian (*testing*) terhadap sebuah sistem komputerisasi di suatu perusahaan ?
4. Apa saja langkah-langkah yang dilakukan untuk memperbaiki kesalahan yang ada di sebuah sistem komputerisasi ?
5. Bagaimana cara anda mengganti sistem komputerisasi lama yang salah dengan sistem komputerisasi baru agar tidak mengganggu proses yang sedang berjalan ?
6. Kapan saja waktunya bahwa sistem komputerisasi harus dicek kevalidannya ?
7. Berapa lama waktu yang diperlukan untuk menjalankan sistem paralel ?
8. Apa saja yang harus disiapkan untuk melakukan *testing* terhadap suatu sistem komputerisasi ?
9. Bila kesalahan sistem komputerisasi dibagi menjadi 2 bagian, yaitu kesalahan *hardware* dan kesalahan *software*, berikan contoh masing-masing jenis kesalahannya.
10. Apa manfaat mendokumentasikan perancangan sistem dalam membantu mendapatkan kesalahan pada suatu perancangan sistem ?

DEMO COPY